PRT582-SOFTWARE ENGINEERING PROCESS AND TOOLS

ASSIGNMENT 1

SOFTWARE UNIT TESTING REPORT

Student Name: Kat Huynh

Student ID: 336951

Professor: Charles Yeo

CHARLES DARWIN UNIVERSITY

COLLEGE OF ENGINEERING, INFORMATION TECHNOLOGY AND ENVIRONMENT

SEPTEMBER 2021

Introduction

This paper is a report on the scrabble score project that is programmed in Python. The project is created using Test Driven Development (TDD) method and unittest automated testing tool. While, automation testing tools are the main drives of TDD method. TDD is a procedure that induce developers to keep writing new codes to pass the failing automated test. It is an effective and sustainable programming approach that helps developers to narrow down the scope of the project. It is indeed necessary to implement TDD and automation testing to this Scrabble Score Project.

Objectives

The project aims to developed a well-functioning Scrabble Score game as per specifications. The Scrabble Score is a word-guessing game run under a 15 seconds timer limit. Player is requested to find a word with letters that meet the amount given. Scores will be calculated accordingly to the table below:

Tabela 1 Point Table

Letter	**Points**
A, E, I, O, U, L, N, R, S, T	1
D, G	2
B, C, M, P	3
F, H, V, W, Y	4
K	5
J, X	8
Q, Z	10

The longer the player takes to come up with the answer the lower the score. Total score will be deducted by 70% after the first 5 seconds, and by 50% after 10 seconds of not receiving a correct answer.

Example:

Find word of 4 letters: BEAD

$t \leq 5$	: 7 points
$5 < t \leq 10$	: 4.89 points
$t > 10$	: 3.5 points

Requirements

A complete Scrabble Score program has to meet these base requirements:

1. Player's input is not case sensitive.
2. Feedback of InvalidCharacter if player is not entering alphabetical symbols.

3. Feed back of InvalidWordLength if player is not entering the correct word length.
4. Feed back of InvalidWord if player is not entering the word that is in the dictionary.
5. Scores are accurately calculated for every finishing time.
6. Random number for word is generated in range (1,24).
7. 15 Seconds timer for the game.

Choosing programming language and automated unit testing tool:

The Scrabble Score game is programmed with Python and the unittest automated testing tool. As a matter of fact, Python consists of many built-in frameworks that make testing easier. With the help of unittest, Test Case implementation can be easily executed with Python as a programming language. Unittest can help to cover a large part of testing without having to manually test the inputs and outputs through the program.

Process

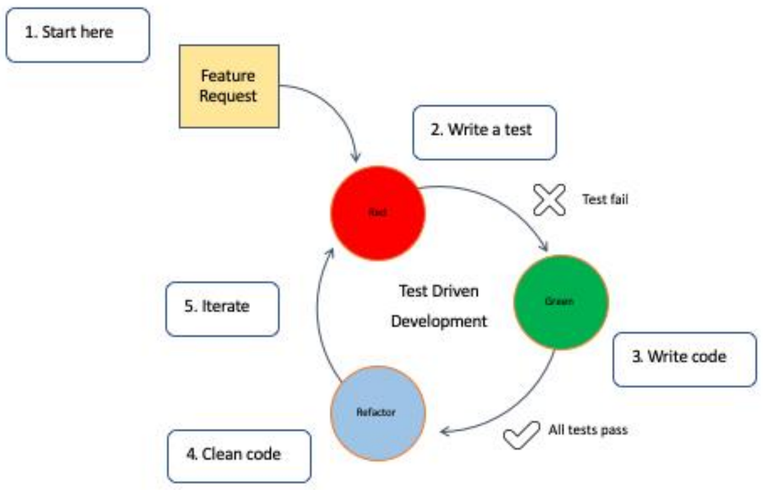


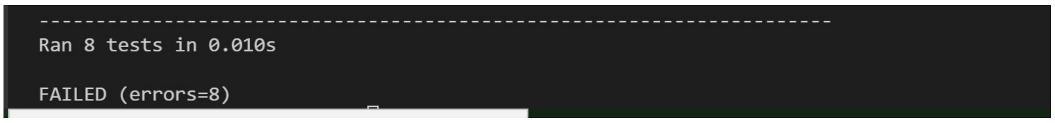
Figura 1 TDD-Flow Chart

The procedure of TDD includes the following steps:

1. Programmers firstly analyze the feature request. In this case, there are 8 feature elements need to be tested. Three out of which are relevant to input type of the word which can be created in one function single-handedly. Regarding requirement 1 and 5, there has to be a proper implementation of Table 1 and the amount of elapsed times. Although the two are quite similar, requirements 5 cover a much more detailed approach. Requirement 1 can be tested under the circumstances that exclude the time factor. As per requirement 4, timer function has to work properly, same for requirements 6 with the random number generator function.

It is noteworthy that requirement 1 is divided into 2 test cases whereby uppercase letters were entered for firstly syntax check of word and secondly word scoring(in the case that word length met the condition- as most length measurements are not case sensitive)

2. Interpret the requirements by creating test cases with unittest. Test file is kept separated from the main program file. The program file is, on the other hand, imported to test file. Initially, there has not been any code written for the Scrabble Score game, requirements should produce all 8 failed tests.

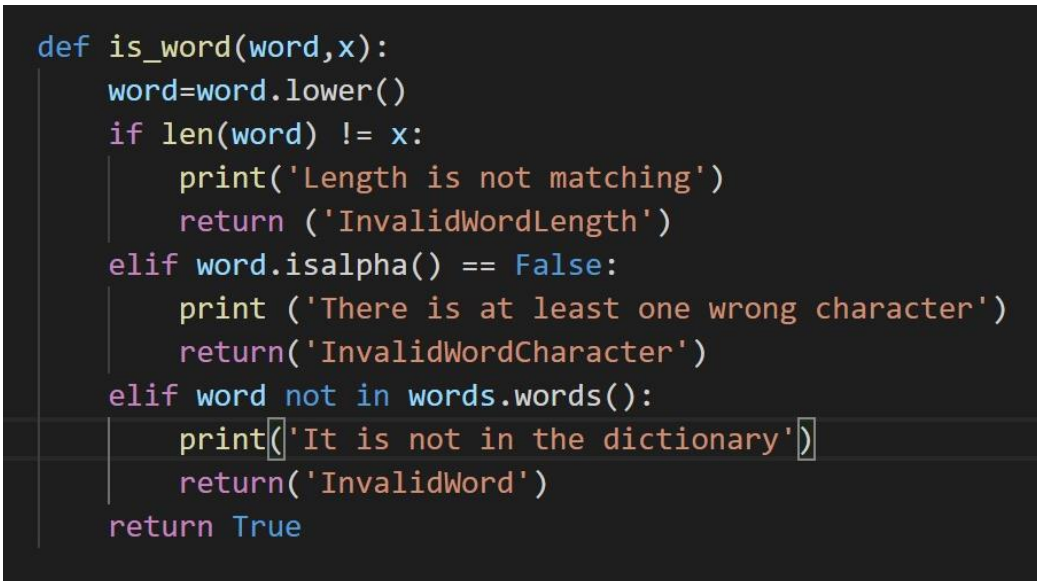


```
----------------------------------------------------------------------
Ran 8 tests in 0.010s

FAILED (errors=8)
```

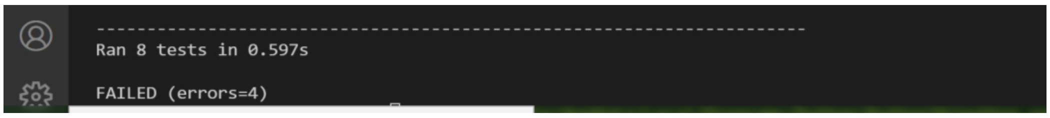
Figura 2 Initial Test Cases Results

3. Repetitive writing and implementing codes until all tests pass. Through Requirements 1,2,3 include of 4 test cases: is_word(word,word_length) function is written to rule out any cases that with invalid word length, invalid word character regardless of word cases.(Refers to picture below)



```python
def is_word(word,x):
    word=word.lower()
    if len(word) != x:
        print('Length is not matching')
        return ('InvalidWordLength')
    elif word.isalpha() == False:
        print ('There is at least one wrong character')
        return('InvalidWordCharacter')
    elif word not in words.words():
        print('It is not in the dictionary')
        return('InvalidWord')
    return True
```

Figura 3 Function is_word

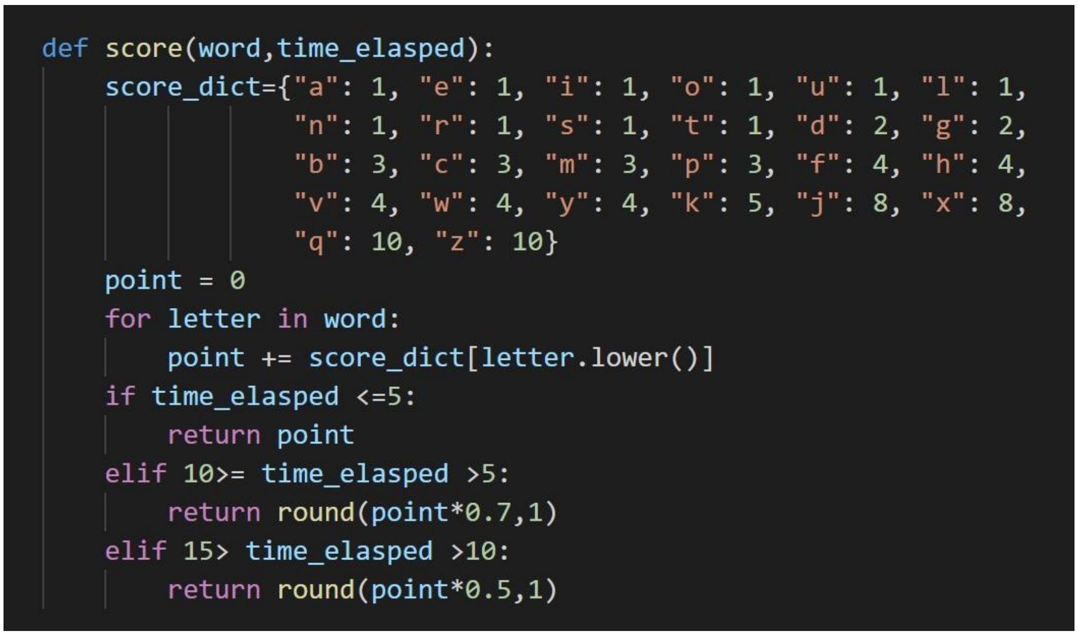


```
----------------------------------------------------------------------
Ran 8 tests in 0.597s

FAILED (errors=4)
```

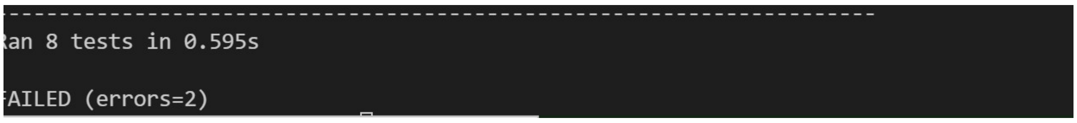
Figura 4 Test Cases Result for Req 1,2,3

To pass requirement 4,5 test cases, scoring function score(word, time_elasped) is used to identify if word is existing in library, and moreover generate scores for different finishing time.



```python
def score(word,time_elasped):
    score_dict={"a": 1, "e": 1, "i": 1, "o": 1, "u": 1, "l": 1,
                "n": 1, "r": 1, "s": 1, "t": 1, "d": 2, "g": 2,
                "b": 3, "c": 3, "m": 3, "p": 3, "f": 4, "h": 4,
                "v": 4, "w": 4, "y": 4, "k": 5, "j": 8, "x": 8,
                "q": 10, "z": 10}
    point = 0
    for letter in word:
        point += score_dict[letter.lower()]
    if time_elasped <=5:
        return point
    elif 10>= time_elasped >5:
        return round(point*0.7,1)
    elif 15> time_elasped >10:
        return round(point*0.5,1)
```

Figura 5 score function

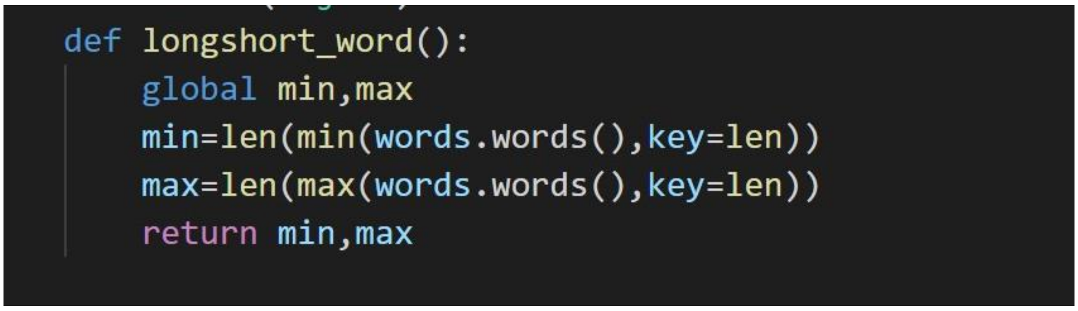


```
----------------------------------------------------------------------
an 8 tests in 0.595s

AILED (errors=2)
```

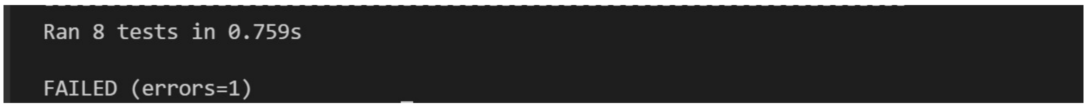
Figura 6 Test Results for Req 4,5

To make sure that randomly generated number are not out of range of the words existing in the library, test case for requirement 6 was written with longshort_word() function to identify the minimum and maximum length of words in the library.



```python
def longshort_word():
    global min,max
    min=len(min(words.words(),key=len))
    max=len(max(words.words(),key=len))
    return min,max
```

Figura 7 longshort_word function

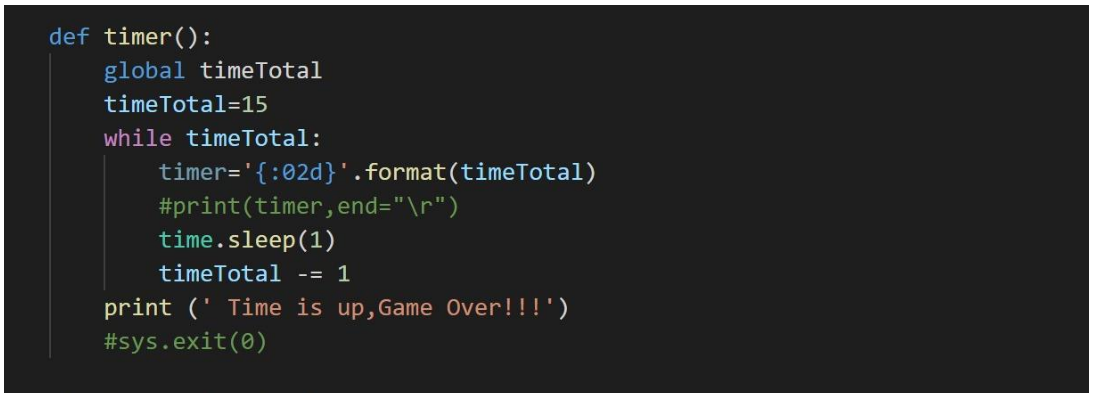


```
Ran 8 tests in 0.759s

FAILED (errors=1)
```

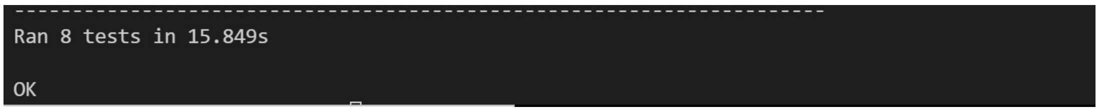
Figura 8 Test Results for Req 6

Finally, to make sure that the game is running in 15 seconds time limit. The function timer() is created which is threaded to the main play function to limit playing time to 15 secs.



```
def timer():
    global timeTotal
    timeTotal=15
    while timeTotal:
        timer='{:02d}'.format(timeTotal)
        #print(timer,end="\r")
        time.sleep(1)
        timeTotal -= 1
    print (' Time is up,Game Over!!!')
    #sys.exit(0)
```

Figura 9 timer function



```
----------------------------------------------------------------------
Ran 8 tests in 15.849s

OK
```

Figure 10 Test Results for Req 7

4. Code Refactoring: All functions are compiled to a workable file. Codes are restructured and refactored to different format that is more functional and effective. Remove comments and redundant codes, replace codes with much more effective commands.
5. Double check codes at last stage, finalize into final product and repeat the cycle as new requirements come up.

Conclusions

The TDD procedure with unittest was executed successfully, all requirements are passed. Although most of the major functional requirements of the program were included in the unit testing cases. There are still certain non functional requirements can be added in as new features (such as the timer display test case, program performance tests,… etc.). These non functional tests that can greatly enhance the player journey, and improve the program to a much more well grounded Scrabble Score game.

.

jamiemoore1997/assign1 (github.com)